# CONTEXT-AWARE REASONING ENGINE WITH HIGH LEVEL KNOWLEDGE FOR SMART HOME

Jiaqi Zhu, Lee Vwen Yen, Jit Biswas
*Institute for Infocomm Research ($I^2R$), Agency for Science, Technology and Research (A*STAR), Singapore, Singapore*

Mounir Mokhtari, Thibaut Tiberghien, Hamdi Aloulou
*Image & Pervasive Access Lab (IPAL), $I^2R$, A*STAR, Singapore, Singapore*

Keywords: Context-aware, Reasoning engine, High level knowledge, Smart home, Sensor network, Pervasive environment, Ambient assistive.

Abstract: We are interested in providing people living or working in smart home environment with sensor network based assistive technology. We propose a novel rule-based reasoning engine that could be used in ubiquitous environments to infer logical consequences from events received over a sensor network. We introduce methods for rule design with high level knowledge input and using minimum information to infer micro-context. Personalised profiles can be introduced into the reasoning engine to customise features for a particular user using our rule refinement and generation module. New mechanism for sensor-engine communication is also introduced. As a proof of concept, a prototype system (using DROOLS) has been developed to demonstrate the functionalities of our reasoning engine in a simulated smart home environment.

## 1 INTRODUCTION

Homes have basically not changed in centuries and the ways we live in homes have not varied much too. However, with the introduction of the Smart Home concept, this interdisciplinary research that addresses the challenges facing the future of home technologies, offers a range of technological benefits to the user (Feki, 2007). Coupled with the gradual trend of ageing populations around the world, much interest have been placed into designing and developing smarter systems to assist and enable an individual to stay at home and take care of themselves independently without the need for constant monitoring of their Activities of Daily Living (ADL).

In current hospitals, nursing homes and other similar health care facilities, nurses and caregivers often have to manage many duties in order to provide excellent patient care. These duties include the need to dispense medication throughout the day at stipulated timings and also having to go on their rounds in their designated wards to observe patients and ensure there are no significant incidents. These two main duties take up quite a lot of the nurses' time. Hence there would be insufficient medical personnel and caregivers that are available to provide prompt and quick responses to every patient around the clock. Emergency situations that require immediate responses have to be given priority over other common occurring incidents and thus assistance could not always be rendered to patients in a timely manner.

The solution to these situations is to install Ambient Intelligent systems and sensors in homes, hospitals and nursing homes to assist nurses and caregivers in monitoring patients or the elderly. However, sensors are only able to provide low-level context which is useful for informing users about simple events like opening or closing a door. Middleware like a reasoning engine would then be used to receive sensor information via a sensor network and also to infer higher-level context like a person's ADL.

This paper looks at an integrated approach consisting of building a rule-based reasoning engine that initially generates generic rules with modifiable and adjustable variables based on a person's profile.

Zhu J., Vwen Yen L., Biswas J., Mokhtari M., Tiberghien T. and Aloulou H..
CONTEXT-AWARE REASONING ENGINE WITH HIGH LEVEL KNOWLEDGE FOR SMART HOME .
DOI: 10.5220/0003396202920297
In *Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems* (PECCS-2011), pages 292-297
ISBN: 978-989-8425-48-5

The reasoning engine could also adapt to different smart home environments through a rule generation process. Rules are created such that logical consequences could be inferred from low-level context received over a sensor network. The reasoning engine is also built to detect any anomalies in a person's ADL and also to reason if any accident has happened. Based on a given set of rules controlled by the nurses or caregivers, alerts or reminders could then be sent out to the medical personnel in-charge or patients whenever abnormalities are detected, for example when somebody stays too long in the shower and needs to be reminded to leave the washroom. With this set of rules, nurses and caregivers would be able to determine and configure actions to be taken and the different type of reminders to be issued based on each person's unique personality or habits.

In scenarios where sensors are unavailable to transmit sensor information due to low battery power or incorrect sensor data being transmitted to the reasoning engine, a probabilistic model would be used to determine the type and number of rules that are to be fired and the subsequent consequences due to the fired rules.

In addition, we note that existing reasoning engines are mostly governed by rules that are pre-defined by programmers before being deployed for end-users usage. Editing of such initial set of rules is often too difficult for non-technical end-users and is also a barrier for user's acceptance towards the system. Therefore we intend to only host generic rules with a pre-defined setting and also develop a control mechanism that allows the user to determine preferences within their personal profiles to be uploaded into our reasoning engine.

The rest of the paper is organised as follows. Existing reasoning engines and related work are listed and compared in Section 2. We introduce our novel context-aware reasoning engine in Section 3. In Section 4, we describe the implementation of our prototype and scenario which our prototype has already been tested in, together with feedback from medical personnel. We end with the conclusion in Section 5.

## 2 RELATED WORK

We are aware of current reasoning engines with similar capabilities pertaining to the uncertainty aspect. Many probabilistic schemes for context processing have been used to tackle the issue of uncertainty. The theory on fuzzy logic (Zadeh, 1996) and Hidden Markov Models (HMM) (Krogh et al., 1994) have been mentioned and discussed as potential probabilistic schemes that could be used (Dargie, 2007). Ranganathan et al. (Ranganathan, 2004) have used Microsoft's Belief Network (MSBN) software to create Bayesian networks to represent relationships between events. Liu Peizhi et al. (Liu, 2008) applied the combination rule in Dempster-Shafer Theory of Evidence (DST) to their reasoning mechanism to construct the inferencer. There was work done on Dynamic Bayesian Networks (DBN) that are able to represent and learn in order to produce more complex models. Murphy modelled hierarchical HMMs as DBNs, as part of his work (Murphy, 2002).

These approaches, apart from fuzzy logic, are mainly generalizations of the Bayesian network and are proposed as possible solutions to solving the uncertainty problem by providing the probabilistic reasoning mechanism. Our approach towards uncertainty is to also create a probabilistic model with probabilistic information optimally pre-defined. However, generic rules related to common smart home scenarios would initially be obtained via a rules repository.

We could not afford to have learning processes when we deploy our reasoning engine as it is expected to be effective on deployment. Therefore, as and whenever the reasoning engine is unable to handle any given situation with the prepared rules, the end-user could personalise or customise the rules and probabilistic information to meet their needs. This manner of customisation would be covered by our rule generation process and end-user control mechanism within our reasoning engine. User profiles from different individuals could also be used for customisation purposes.

There are other reasoning engines that are rule based, but do not take into account high level knowledge that could be used to optimize rule design (Goh, 2007). There are also methods that utilise high level knowledge but are instead used to computerize Clinical Practice Guidelines (CPG) so as to operationalize them within Clinical Decision Support Systems (Hussain, 2007).

## 3 OUR REASONING ENGINE

### 3.1 Rule with High Level Knowledge

Our context-aware reasoning engine (later referred to as "engine") is rule driven and based on first order logic. It is able to work without training. The engine uses predefined rules in a rule repository (in our case about 30 rules) as the initial rules. The initial rules are predefined in the way that domain expert knowledge is built into them. Domain expert knowledge could be the knowledge from doctors or nurses in the domain of healthcare as they know exactly how a patient or an elderly performs his/her activities of daily living. The knowledge can then be mapped to some rules or components of some specific rules. The components of a standard rule in the rule repository are described in Table 1.

Table 1: Rule components.

| Rule name |
|---|
| Person |
| Location |
| Time |
| Sensor$_1$ status |
| Sensor$_2$ status |
| ... |
| Sensor$_n$ status |
| Sensor selection scheme |
| Firing condition |

The first 4 components are easy to understand. The sensors status is the related sensors behaviour that would imply the underlying micro-context. We have created the "sensor selection scheme". This spirit in the rule design of the engine is that we always use minimum information possible to make judgement of the current context (location of a person, status of location) or the micro-context (activity of a person) although there is more information available. Only when we do not have sufficient information on hand to make a sound judgement, do we actually resort to seek for additional available information to perform inference. This is to alleviate the workload of the processing server and to save on data transmission bandwidth. Firing condition specifies the input conditions of the first order logic.

For example, a rule which monitors person A's showering activity and alerts when person A is showering for too long is defined in Table 2. The duration of sensor$_1$ and sensor$_2$ in active state being the threshold for judging is set according to expert knowledge. Though we have four sensors installed, we try to only use sensor$_1$ and sensor$_2$ in the first place. Only when sensor$_2$ is not available (may due to malfunction or battery failure) or if certain conditions are not satisfied, do we take into account information received from sensor$_3$ and sensor$_4$.

So in the afternoon, when the status of sensor 1 and 2 are met or those of sensor 1, 3 and 4 are met, the showering for too long rule is fired and an alert will be sent to a speaker in the washroom telling person A to finish shower quickly.

Table 2: Rule for showering for too long.

| Rule name | showering for too long |
|---|---|
| Person | A |
| Location | washroom |
| Time | afternoon |
| Sensor$_1$ status | shower pipe shake sensor: active for 15 minutes |
| Sensor$_2$ status | shower area PIR sensor: active for 15 minutes |
| Sensor$_3$ status | shower soap dispenser: pressed |
| Sensor$_4$ status | washroom door reed switch sensor: closed |
| Sensor selection scheme | check sensor 1 and 2 first, and then sensor 3 and 4 |
| Firing condition | if sensor 1 and 2 are satisfied, or if sensor 1, 3 and 4 are satisfied |

## 3.2 Rule Generation and Refinement

The engine uses predefined initial rules obtained from a rules repository. However, we have designed algorithms for the engine to refine the rules accordingly overtime, and generate new rules when necessary. The idea is shown in Figure 1.

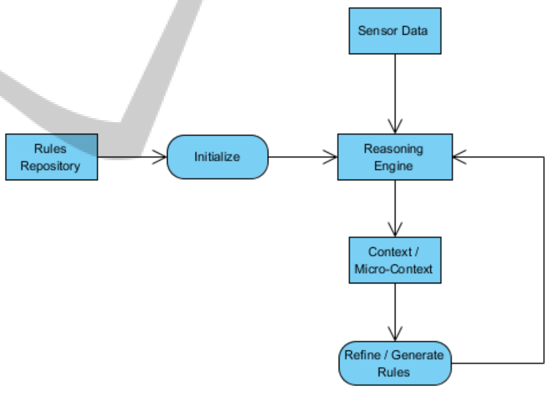

Figure 1: Rule refinement and generation.

Figure 1 shows that the rules repository is used to initialize the reasoning engine. On receiving sensor data, the engine does reasoning to infer the context and micro-context.

The temporal information in the context and micro-context is used to modify the components of the existing rules, such as sensor duration and status. If some new micro-context is discovered, new rules can be generated by combining the spatial, temporal and sensor information.

The engine will save the modified rules and the newly generated rules back into the rules repository so that the repository always reflects the most updated set of rules.

## 3.3 Engine-sensor Communication

One important characteristic of the engine is its Sensor Universal Plug and Play (SUPnP) feature. This feature is to overcome the drawback of existing sensor network smart home systems as it allows the engine to automatically recognize newly deployed sensors and then adopt them to work together with our rules. No restart or re-configure of the engine is needed.

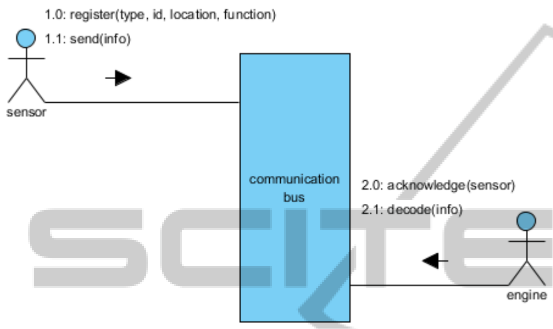

Figure 2: SUPnP feature.

When a new sensor is deployed into the network, it has to use a "register" function (provided by the engine) to register itself. The sensor sends its information such as the sensor type, identity (must be unique), location and functionality to the engine. The engine then adds the sensor into its working sensor list and uses the 'acknowledge' function to reply. Subsequently, the sensor and the engine communicate through a communication bus using 'send' and 'decode' functions respectively.

With the design of SUPnP, the engine will no longer need to stop and restart whenever a new sensor comes. When used in conjunction with our rule generation and refinement module, the designer of the engine does not have to manually re-write existing rules or write new rules in order to include the sensors into the system. The system will automatically add new sensors into rules according to their functionality using their unique sensor id.

## 3.4 Other Features

Some other features of the engine include abstract rule, rule image and rule verification.

An abstract rule is a rule with one or more components being some variable(s). A rule image is an instantiation of an abstract rule. When we want to create a rule for multiple people and it has some component with values that will vary for different people, we would firstly create an abstract rule with the component being a variable instead of a value. We then retrieve the necessary related value from a profile and replace the variable with a specific value specifically denoted for the particular person. So, this rule becomes an image of the abstract rule. With the introduction of the rule image, we do not have to explicitly define different rules for different people. The engine will create an image of an abstract rule the moment it receives an input from a user's profile. Abstract rules are also stored in the rules repository (refer to Section 3.1). Figure 3 illustrates how this works.

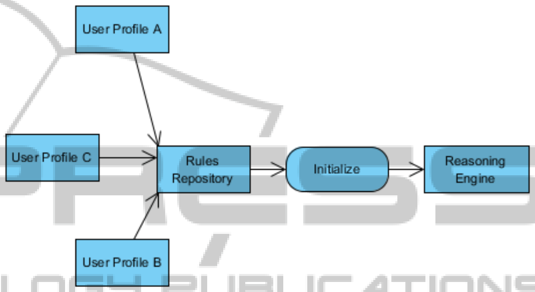

Figure 3: User profile is used to instantiate abstract rules.

We are using Process Analysis Toolkit (PAT) (PAT, 2010) to conduct verification of rules, i.e. to discover conflicting, redundant and non-reachable rules. This is done based on PAT's model checking techniques.

## 4 PROTOTYPE

We used a Rules Engine implementation termed JBoss rules (Jboss, 2010), tailored for Java language to assist us in creating the rule file. This rule file can be edited with normal text editors or IDEs like Eclipse or NetBeans. By using DROOLS, we also have the ability to use Domain Specific Language (DSL) to simplify our rules for end-users to understand.

The sequence flow of our prototype reasoning engine starts off with the administrator generating and obtaining the initial set of rules from the rules repository. If there are occasions where specific new rules need to be constructed or present rules need to be modified, the amendment could be completed by the administrator. Using a main java program MainReasoner to read in the rules, the reasoner would accept user profiles and using the information from the profile to create rule images, which are instantiations of the abstract rules. In this way, the reasoner would be able to personalise particular sets of rules for users according to their profiles. The modified rule image would then be inserted into the working memory of the knowledge base.

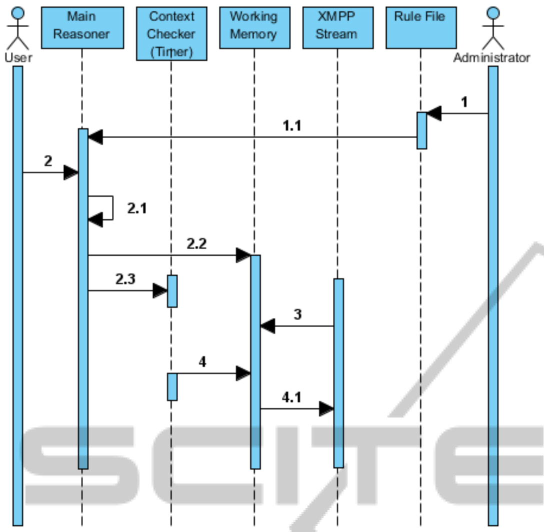

Figure 4: Sequence diagram for our prototype reasoning engine with numbered annotations as follows:

1 setRule()
    1.1 readRules()
2 sendProfile()
    2.1 initialiseUser()
    2.2 insertRules()
    2.3 setTimer()
3 decode()
4 fireAllRules()
    4.1 sendData()

Sensor information in string format would be transmitted by sensors and received over the XMPP (XMPP, 2008) stream whenever an event has occurred. The decoding of the strings would then provide us with information of events, which would then be used for comparison with the current information within the working memory. This updating process would also allow us to decide whether the conditions of the rules within the working memory are fulfilled and if any appropriate alerts or reminders need to be fired.

A context checker which works as a timer would constantly check the context at every fixed interval, detecting if there are any changed states or variables within the working memory. If the XMPP stream is silent and there are no events within the fixed interval, the context checker would then proceed to check through the rules within the working memory and decide on any alerts or reminders based on the fulfilled conditions of rules within the working memory.

The alerts would be in the form of sending messages back into the XMPP stream whereby a separate UI component would then pick up the message and translate them into verbal reminders to be communicated to the users involved.

Working together with two other teams consisting of a sensor team and a UI team, we were able to present several demonstrations of our engine's capabilities to medical personnel and caregivers. We have generated a few scenarios based on inputs and feedback from medical institutions about the current situation regarding elderly and patient care, so as to allow our scenarios to be as closely related to real-life situations as possible. The general feedback obtained from these personnel was largely positive with the view that this system in conjunction with our sensors would be suitable for future clinical trials as the capabilities of the entire system proved to be useful for nurses or caregivers, and would also help in reducing their workload during their shifts.

It was also pointed out that present Ambient Intelligent systems tend to over-saturate the user with too many reminders when an incident or user error has occurred, which we took into consideration when building our system. Hence, with further considerable information from medical personnel and users, we would be able to further fine-tune our engine to cater to their actual needs and provide a better service.

## 5 CONCLUSIONS

In this paper, we have presented our context-aware reasoning engine for ambient assistive systems. The engine does not require training as it can start working with predefined rules in a rule repository. The engine infers micro-context from minimum information possible and then uses the micro-context to refine its rules or generate new rules. It communicates with sensors in an automated way that new sensors will be incorporated into the engine without the need to restart the engine.

A DROOLS version has already been implemented and demonstrated. Currently, we are developing a new version of the engine based on ontology using JENA (JENA, 2010). We plan to make it OSGi (OSGi, 2010) ready, so other modules in the ambient assistive system will be able to call methods implemented in the engine directly. Our work is also part of an ongoing project on Activity Monitoring and UI Plasticity for supporting Ageing with mild Dementia at Home (AMUPADH) (Biswas, 2010).

Some of the future works include dealing with uncertainty from sensors and managing the reliability of the rule factory when the number of rules is large.

## REFERENCES

Feki, M. A., 2007. *An Ontology based Context aware modelling and reasoning to enhance Human Environment Interaction.* National Institute of Telecommunication.

Zadeh, L. A., 1996. *Fuzzy Sets, Fuzzy Logic, and Fuzzy Systems: Selected Papers.* World Scientific Publishing Co., Inc.

Krogh A., Brown, M., Mian, I. S., Sjolander K., and Haussler D., 1994. *Hidden Markov models in computational biology: Applications to protein modelling.* Journal of Molecular Biology.

Dargie, W., 2007. *The Role of Probabilistic Schemes in Multisensor Context-Awareness.* In *Fifth IEEE International Conference on Pervasive Computing and Communications Workshops (PerComW'07).*

Ranganathan, A., Al-Muhtadi, J., Campbell, R. H., 2004. *Reasoning about Uncertain Contexts in Pervasive Computing Environments.* In *IEEE Pervasive Computing*.

Liu, P., Zhang, J., 2008. *A Context-Aware Application Infrastructure with Reasoning Mechanism Based on Dempster-Shafer Evidence Theory.* In *Vehicular Technology Conference (VTC Spring 2008).*

Murphy, K. P., 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning.* University of Carlifornia, Berkeley.

Goh E., Chieng D., Mustapha A. K., Ngeow Y. C., and Low H. K., 2007. *A Context-Aware Architecture for Smart Space Environment*. In *International Conference on Multimedia and Ubiquitous Engineering.*

Hussain S., and Abidi S. S. R., 2007. *Ontology Driven CPG Authoring and Execution via a Semantic Web Framework*. In *The 40th Hawaii International Conference on System Sciences.*

Biswas, J., Mokhtari, M., Dong, J. S., Yap, P., 2010. *Mild Dementia at Home – Integrating Activity Monitoring, User interface plasticity and Scenario verification.* In *8th International Conference on Smart Homes and Health Telematics, (ICOST 2010).*

Jboss Rules (Drools 5.1), *http://www.jboss.org/drools/documentation.html* (accessed October 2010).

PAT: Process Analysis Toolkit, *http://www.comp.nus.edu.sg/~pat* (accessed December 2010).

JENA, Semantic Web Framework for Java, *http://jena.sourceforge.net* (accessed December 2010).

Extensible Messaging and Presence Protocol, XMPP, *http://www.e-framework.org/Default.aspx?tabid=708* (accessed December 2010).

Open Services Gateway initiative (OSGi), *http://www.osgi.org/Main/HomePage* (accessed December 2010).